The Patent Office
Concept House
Cardiff Road
Newport
South Wales
NP10 8QQ

I, the undersigned, being an officer duly authorised in accordance with Section 74(1) and (4) of the Deregulation & Contracting Out Act 1994, to sign and issue certificates on behalf of the Comptroller-General, hereby certify that annexed hereto is a true copy of the documents as originally filed in connection with the patent application identified therein.

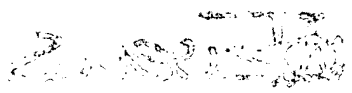
In accordance with the Patents (Companies Re-registration) Rules 1982, if a company named in this certificate and any accompanying documents has re-registered under the Companies Act 1980 with the same name as that with which it was registered immediately before re-registration save for the substitution as, or inclusion as, the last part of the name of the words "public limited company" or their equivalents in Welsh, references to the name of the company in this certificate and any accompanying documents shall be treated as references to the name with which it is so re-registered.

In accordance with the rules, the words "public limited company" may be replaced by p.l.c., plc, P.L.C. or PLC.

Re-registration under the Companies Act does not constitute a new legal entity but merely subjects the company to certain additional company law rules.

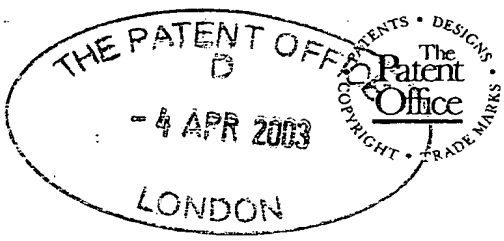
Signed J Evans.

Dated 23 December 2003



Patents Form 7/77

Patents Act 1977
(Rule 15)



Statement of inventorship and of right to grant of a patent

The Patent Office

Cardiff Road
Newport
South Wales
NP10 8QQ

1. Your reference — P015899GB

2. Patent application number *(if you know it)* — 0307821.9 — 04 APR 2003

3. Full name of the or of each applicant — ARM Limited

4. Title of the invention — Controlling Execution of a Block of Program Instructions within a Computer Processing System

5. State how the applicant *(s)* derived the right from the inventor *(s)* to be granted a patent — By virtue of employment

6. How many, if any, additional Patents Forms 7/77 are attached to this form? *(see note (c))* — 0

7. I/We believe that the person *(s)* named over the page *(and on any extra copies of this form)* is/are the inventor *(s)* of the invention which the above patent application relates to.

Signature  Date 4 April 2003

D Young & Co (Agents for the Applicants)

8. Name and daytime telephone number of person to contact in the United Kingdom — Nigel Robinson — 023 8071 9500

Notes

a) If you need help to fill in this form or you have any questions, please contact the Patent Office on 08459 500505.

b) Write your answers in capital letters using black ink or you may type them.

c) If there are more than three inventors, please write the names and addresses of the other inventors on the back of another Patents Form 7/77 and attach it to this form.

d) When an application does not declare any priority, or declares priority from an earlier UK application, you must provide enough copies of this form so that the Patent Office can send one to each inventor who is not an applicant.

e) Once you have filled in the form you must remember to sign and date it.

Enter the full names, addresses and postcodes of the inventors in the boxes and underline the surnames

Surname: VASEKIN

First Names: Vladimir

76 Winfold Road
Waterbeach
Cambridge
CB5 9QF

Patents ADP number *(if you know it)*: 8604068001

Surname:

First Names:

Patents ADP number *(if you know it)*:

Surname:

First Names:

Patents ADP number *(if you know it)*:

Reminder

Have you signed the form?

Patents Form 1/77

Patents Act 1977
(Rule 16)

THE PATENT OFFICE
D
- 4 APR 2003

PATENTS • DESIGNS • COPYRIGHT • TRADE MARKS
The Patent Office

Request for grant of a patent

(See the notes on the back of this form. You can also get an explanatory leaflet from the Patent Office to help you fill in this form)

The Patent Office

Cardiff Road
Newport
South Wales
NP10 8QQ

1. Your reference — P015899GB

07APR03 E797916-1 D02246
P01/7700 0.00-0307821.9

2. Patent application number
(The Patent Office will fill in this part) — 0307821.9

0 4 APR 2003

3. Full name, address and postcode of the or of each applicant *(underline all surnames)*

ARM Limited
110 Fulbourn Road
Cherry Hinton
Cambridge
CB1 9NJ

Patents ADP number *(if you know it)* 7498124002

If the applicant is a corporate body, give the country/state of its incorporation — UK

4. Title of the invention

Controlling Execution of a Block of Program Instructions within a Computer Processing System

5. Name of your agent *(if you have one)*

D Young & Co

"Address for service" in the United Kingdom to which all correspondence should be sent *(including the postcode)*

21 New Fetter Lane
London
EC4A 1DA

Patents ADP number *(if you know it)* 59006

6. If you are declaring priority from one or more earlier patent applications, give the country and the date of filing of the or of each of these earlier applications and *(if you know it)* the or each application number

Country	Priority application number *(if you know it)*	Date of filing *(day / month / year)*

7. If this application is divided or otherwise derived from an earlier UK application, give the number and the filing date of the earlier application

Number of earlier application	Date of filing *(day / month / year)*

8. Is a statement of inventorship and of right to grant of a patent required in support of this request? *(Answer 'Yes' if:*

a) any applicant named in part 3 is not an inventor, or
b) there is an inventor who is not named as an applicant, or
c) any named applicant is a corporate body.
See note (d))

Yes

Patents Form 1/77

9. Enter the number of sheets for any of the following items you are filing with this form. Do not count copies of the same document

Continuation sheets of this form	0
Description	10
Claim *(s)*	6
Abstract	1
Drawing *(s)*	4

10. If you are also filing any of the following, state how many against each item.

Priority documents	0
Translations of priority documents	0
Statement of inventorship and right to grant of a patent *(Patents Form 7/77)*	2
Request for preliminary examination and search *(Patents Form 9/77)*	1
Request for substantive examination *(Patents Form 10/77)*	0
Any other documents *(please specify)*	0

11. I/We request the grant of a patent on the basis of this application.

Signature D Young & Co (Agents for the Applicants) Date 4 April 2003

12. Name and daytime telephone number of person to contact in the United Kingdom: Nigel Robinson 023 8071 9500

Warning

After an application for a patent has been filed, the Comptroller of the Patent Office will consider whether publication or communication of the invention should be prohibited or restricted under Section 22 of the Patents Act 1977. You will be informed if it is necessary to prohibit or restrict your invention in this way. Furthermore, if you live in the United Kingdom, Section 23 of the Patents Act 1977 stops you from applying for a patent abroad without first getting written permission from the Patent Office unless an application has been filed at least 6 weeks beforehand in the United Kingdom for a patent for the same invention and either no direction prohibiting publication or communication has been given, or any such direction has been revoked.

Notes

a) *If you need help to fill in this form or you have any questions, please contact the Patent Office on 08459 500505.*

b) *Write your answers in capital letters using black ink or you may type them.*

c) *If there is not enough space for all the relevant details on any part of this form, please continue on a separate sheet of paper and write "see continuation sheet" in the relevant part(s). Any continuation sheet should be attached to this form.*

d) *If you have answered 'Yes' Patents Form 7/77 will need to be filed.*

e) *Once you have filled in the form you must remember to sign and date it.*

f) *For details of the fee and ways to pay please contact the Patent Office.*

CONTROLLING EXECUTION OF A BLOCK OF PROGRAM INSTRUCTIONS WITHIN A COMPUTER PROCESSING SYSTEM

This invention relates to the field of data processing systems. More particularly, this invention relates to the control of execution of a block of program instructions within a data processing system.

It is known that computer programs often contain sequences of program instructions that are frequently repeated within the computer program. In order to produce a computer program with a smaller code size, it is known to arrange such blocks of computer program instructions into functions or subroutines which can be called from various positions within the computer program.

It is normal for such subroutines to terminate with a return instruction which commands the data processing apparatus to return to the instruction immediately following the point in the computer program from where the call to the subroutine was made. When the subroutine or block of instructions is short in length, then the overhead of providing a return instruction at the end of the subroutine can form a significant proportion of the size of the subroutine itself. As an example, if the subroutine block of program instructions being called is only three instructions in length, then the necessary return instruction at the end of the block increases this length to four instructions and results in a significant increase in code size when this is repeated across a large number of such subroutines which may be included within a computer program as a whole.

It is also known to provide data processing systems including a dictionary function whereby an instruction in the program is a dictionary instruction which triggers a reference to be made to a stored dictionary table where there is a pointer to a memory location storing a sequence of program instructions to be executed in response to that dictionary function. The dictionary table may also include an indication of the length of that block of instructions. The dictionary table approach has the disadvantage that an additional memory construct, namely the dictionary table, needs to be provided within the data processing system as well as additional

registers for keeping track of full length memory addresses for the dictionary instruction and the position within the block of program instructions called. In the context of blocks of program instructions which are very short in length, the storage requirements of the dictionary table entries relating to those small blocks of program instructions form a significant proportion of the storage requirements for those blocks of instructions in a manner which is disadvantageous.

A further disadvantage of the dictionary table approach is that it is a more radical change to an existing data processing system architecture if it is to be added to such an existing data processing system architecture. It is desirable that new functionality should be added to a data processing system architecture in a manner which minimises the degree of architectural change necessary.

It is also known within VAX architecture computers to provide an execute instruction which commands the system to execute an instruction found at a memory location specified by the execute instruction. This type of operation can be considered as a one-for-one replacement of the execute instruction within the program code by different instructions pointed to by those execute instructions. This type of functionality is particularly useful for debugging and diagnostic purposes but does not yield significant code density improvements.

Viewed from one aspect the present invention provides apparatus for processing data, said apparatus comprising:

an instruction fetching circuit operable to fetch program instructions from a sequence of memory locations;

an instruction decoder responsive to program instructions fetched by said instruction fetching circuit to control data processing operations specified by said program instructions; and

an execution circuit operable under control of said instruction decoder to execute said data processing operations; wherein

said instruction decoder is responsive to an execute block instruction to trigger fetching of a block of two or more program instructions by said instruction fetching circuit and execution of said block of two or more program instructions by said execution circuit, said block of two or more instructions containing a number of

program instructions specified by a block length field within said executed block instruction and being stored at a memory location specified by a location field within said execute block instruction.

The present technique recognises that for a large number of blocks of program instructions that can advantageously be the subject of calls from different points within a program, the return instruction represents a significant overhead. Combined with this is the realisation that such small blocks of program instructions rarely need to include branch instructions such that when they are started they will with a high degree of probability always be run to their conclusion, i.e. result in a fixed number of program instructions being fetched and executed. Accordingly, the execute block instruction provided by the present technique specifies within the execute block instruction both the location of the block of program instructions to be executed as well as the length of that block of program instructions. Accordingly, there is no need for the block of program instructions to include a return instruction, since the length of the block is already known as specified within the execute block instruction and the return to the main program can be triggered when the final instruction within the block of program instructions has been executed. This execute block instruction extends the advantages of program instruction calls to small blocks of program instructions. The technique is also particularly well suited to use by program compilers which can identify frequently occurring small blocks of instructions within a program image and replace these by execute block instructions. The occurrence of a block in the normal code can be used as the target of branch instructions without the need to separately store the block of instructions elsewhere.

It will be appreciated that whilst it is possible that a block of program instructions being called could include a branch overriding the action of the execute block instruction and any return calculated from the length of the block specified in the execute block instruction, preferred embodiments to utilise the length of the block as specified in the execute block instruction to trigger a return to a program instruction outside of the block of program instructions once the execution of the block of program instructions has completed.

Whilst the return could be made to a variety of different program locations, such as specified in a final instruction of the block of program instructions, it is normal and advantageous that the return should be made by default to a program instruction immediately following the execute block instruction within the sequence of memory locations storing the main computer program.

The location field within the execute block instruction may specify the location of the block of program instructions in a variety of different ways, such as an absolute address value that advantageously uses an offset field as this is typically more space efficient and can be embedded within the size of bit field available within the execute block instruction.

Preferred embodiments of the invention use a program counter to store an address of a memory location of a program instruction being executed together with a block counter register storing a block count value indicative of a location of a program instruction being executed within a called block of program instructions. In this type of arrangement, when a call is made to a block of program instructions, the program counter register stores the memory location of the execute block instruction whilst the block counter register stores a block counter value indicative of the position within the block of program instructions that has been reached.

The use of a block count register and a program counter register in the above way is particularly useful during exception handling whereby an exception routine can be triggered and a return made after exception handling to the point within the block of instructions that was interrupted in dependence upon the program counter register referencing the execute block instruction and then reference to the block counter value to find the point reached within the block of program instructions.

Viewed from another aspect the present invention provides a method for processing data, said method comprising the steps of:

fetching program instructions from a sequence of memory locations with an instruction fetching circuit;

controlling data processing operations specified by said program instructions with an instruction decoder; and

executing said data processing operations with an execution circuit controlled by said instruction decoder; wherein

said instruction decoder is responsive to an execute block instruction to trigger fetching of a block of two or more program instructions by said instruction fetching circuit and execution of said block of two or more program instructions by said execution circuit, said block of two or more instructions containing a number of program instructions specified by a block length field within said executed block instruction and being stored at a memory location specified by a location field within said execute block instruction.

Viewed from a further aspect the present invention provides a computer program product including a computer program operable to control a data processing apparatus having an instruction fetching circuit operable to fetch program instructions from a sequence of memory locations, an instruction decoder responsive to program instructions fetched by said instruction fetching circuit to control data processing operations specified by said program instructions, and an execution circuit operable under control of said instruction decoder to execute said data processing operations; said computer program including one or more an execute block instructions operable to trigger fetching of a block of two or more program instructions by said instruction fetching circuit and execution of said block of two or more program instructions by said execution circuit, said block of two or more instructions containing a number of program instructions specified by a block length field within said executed block instruction and being stored at a memory location specified by a location field within said execute block instruction.

It will be appreciated that the computer program product could take a wide variety of different forms, such as a storage medium bearing the computer program or a download or transmission of a computer program. The computer program contains one or more executed block instructions of the type discussed above.

Embodiments of the invention will now be described, by way of example only, with reference to the accompanying drawings in which:

Figure 1 schematically illustrates a data processing apparatus of a type suitable for executing execute block instructions;

Figure 2 schematically illustrates a call by an execute block instruction;

Figure 3 is a flow diagram schematically illustrating the execution of a block of program instructions in a non-pipelined environment;

Figure 4 schematically illustrates the interruption of a block of program instructions that has been called; and

Figure 5 schematically illustrates the architecture of a general purpose computer which may executed a computer program using the above techniques.

Figure 1 shows a data processing apparatus 2 including a register bank 4, a multiplier 6, a shifter 8, an adder 10, an instruction pipeline 12, an instruction decoder 14, a prefetch unit 16, a program counter register 18 and an interrupt controller 20. It will be appreciated that the data processing apparatus 2 as illustrated in Figure 1 will typically include many further circuit elements, but these have been omitted for the sake of clarity. In operation, instructions are fetched from a memory under control of the prefetch unit 16 and a memory location as specified in the program counter register 18 into the fetch stage of the instruction pipeline 12. The instructions progress along the instruction pipeline 12 to a decode stage and then to an execute stage in accordance with normal microprocessing techniques. The instruction decoder 14 decodes the program instructions in the decode stage and generates control signals which are used to configure the circuit elements, such as the register bank 4, the multiplier 6, the shifter 8 and the adder 10, to perform specified data processing operations. The register bank 4, the multiplier 6, the shifter 8 and the adder 10 can be considered to be an execute circuit for executing processing operations as specified by program instructions and under control of the control signals generated by the instruction decoder 14. The interrupt controller 20 is responsive to interrupt signals irq to interrupt normal processing activity and trigger execution of an exception handling routine. The interrupt controller 20 forces the prefetch limit 16 to start fetching instructions from the start of the exception handling routine. Upon

completion of the exception handling routine, the previous processing is resumed with a restore being made of the processor state at the point at which the interrupt occurred.

In accordance with the present technique, an execute block instruction is added which specifies an offset address (in this example a negative offset to the current program count value) to a block of program instructions to be fetched. The block has a length specified within the execute block instruction (e.g. up to 16 instructions as specified by a 4-bit field). The instruction decoder 14 has a block counter register 22 added to it to keep track of the number of instructions from the called block of instructions that have been executed so that when the end of that block has been reached the prefetch unit 16 can be commanded to restart fetching instructions from a memory location immediately following the execute block instruction within the initial program flow.

Figure 2 illustrates an area of main program code 24 which is being sequentially executed. The program instructions are fetched in turn by the prefetch unit 16 to the instruction pipeline 12 where they are decoded and then executed. The program counter register 18 keeps a track of the address value corresponding to the program instruction currently being executed. In this example the program instructions are 32-bit instructions and accordingly the program counter value increments between instructions in steps of four bytes.

An execute block instruction (EMB: Execute Macroblock Instruction) 26 is located within the main code 24 at a PC value of "x". This execute block instruction includes within it as fields an offset value being a negative memory offset to a starting location of a reference block of program instructions 28 together with a length field specifying the number of program instructions (y-1) within the block of program instructions 28. The instruction decoder 14 is responsive to the execute block instruction 26 to start fetching program instructions, using the prefetch unit 16, from the new memory location pointed to by the offset field. These instructions are then executed. During execution of this block of program instructions the program counter value stored within the program counter register 18 is not incremented but is instead held at the value corresponding to the execute block instruction 26 itself, namely (x). The block counter register 22 is incremented by the instruction decoder starting from

a value of zero up to a value of (y-1) corresponding to the end of the block of program instructions 28. When the instruction decoder 14 detects that the block counter value has reached a value of (y-1) which matches the length of the block as originally specified within the execute block instruction 26, the instruction decoder 14 then forces the return to the instruction immediately following the execute block instruction 26 by controlling the prefetch unit 16 to fetch that instruction (at location x+4) into the instruction pipeline 12.

As an alternative to the normal return behaviour, it is possible that a branch instruction may be embedded within the block of program instructions 28 directing a branch to an instruction at another memory location. If such a branch occurs, then it serves to clear the pending execute block instruction behaviour and processing proceeds starting from the target of the branch instruction in the normal way. It is also possible to have such a branch instruction as the last instruction within the block of program instructions 28 to trigger a return to a point in the main program code 24 other than the instruction immediately following the execute block instruction 26.

Figure 3 is a flow diagram schematically illustrating the behaviour in a non-pipeline system (i.e. not the pipeline system of Figure 1) of a data processing system responsive to an execute block instruction. At step 30 the instruction decoder detects whether an execute block instruction has been received. When such an execute block instruction has been received, processing proceeds to step 32 at which the address offset value and block length are read from the execute block instruction. At step 34 the first instruction from the block of program instructions pointed to by the current microPC (block counter register 22) is loaded into the system for decoding an execution starting from the memory location pointed to by the current program counter value minus the specified offset value. At step 36 the current program instruction from the block of program instructions is executed.

It will be appreciated that an external interrupt, as will be discussed later, can occur before the step 36 or an internal interrupt during the step 36. Whilst the occurrence of such external and internal interrupts is conventional, the way in which the return location after interrupt servicing is tracked is different when using the present technique and will be discussed in relation to Figure 4.

After step 26, the block counter value is incremented at step 38 and then step 40 determines whether the last instruction within the block of program instructions has yet been reached. If the last instruction has not yet been reached, then processing proceeds to step 42 at which the next instruction is loaded into the system and a return is made to step 36. If the last instruction has been reached, then processing proceeds to step 44 at which a return is made to the main program code in which the execute block instruction occurred at a location immediately following that execute block instruction and the microPC value is set to zero.

Figure 4 schematically illustrates the occurrence of an interrupt during execution of a block of program instructions called by an execute block instruction. Firstly, an execute block instruction occurs at point 46 triggering the start of execution of the block of program instructions at point 48. During the execution of the block of program instructions, the program counter value is held at a value corresponding to the execute block instruction whilst the block counter value is incremented to indicate the position within the block of program instructions concerned. An interrupt occurs during the execution of the block of program instructions and this triggers the start of execution of interrupt handling code at step 50. The state of the data processing apparatus 2 is saved for later restarting by saving the program counter value and the block counter value, as well as other state variables in the normal way. When the interrupt handling code has finished, these saved program counter values and block counter values are restored. The program counter values can be saved in accordance with the normal exception handling mechanisms of processors such as the ARM processors designed by ARM Limited of Cambridge, England. The block counter value can be saved in the context of such processors by using a bit field within the PSR register to be saved as a saved program status register configuration parameter. If the blocks of program instructions that may be called using an execute block instruction are restricted in length to a maximum size of 16 instructions, then the block counter value need only be four bits in length, which can be conveniently represented by a small bit field within the PSR register of an ARM processor.

Termination of the interrupt handling code at point 52 serves to trigger the restoring of the program counter value and the block counter value and resumption of

the block of program instructions at point 54. After the block of program instructions has completed execution at point 56, a return is made to the main program code at point 58.

Figure 5 schematically illustrates a general purpose computer 200 of the type that may be used to implement the above described techniques. The general purpose computer 200 includes a central processing unit 202, a random access memory 204, a read only memory 206, a network interface card 208, a hard disk drive 210, a display driver 212 and monitor 214 and a user input/output circuit 216 with a keyboard 218 and mouse 220 all connected via a common bus 222. In operation the central processing unit 202 will execute computer program instructions that may be stored in one or more of the random access memory 204, the read only memory 206 and the hard disk drive 210 or dynamically downloaded via the network interface card 208. The results of the processing performed may be displayed to a user via the display driver 212 and the monitor 214. User inputs for controlling the operation of the general purpose computer 200 may be received via the user input output circuit 216 from the keyboard 218 or the mouse 220. It will be appreciated that the computer program could be written in a variety of different computer languages. The computer program may be stored and distributed on a recording medium or dynamically downloaded to the general purpose computer 200. When operating under control of an appropriate computer program, the general purpose computer 200 can perform the above described techniques and can be considered to form an apparatus for performing the above described technique. The architecture of the general purpose computer 200 could vary considerably and Figure 5 is only one example.

CLAIMS

1. Apparatus for processing data, said apparatus comprising:

an instruction fetching circuit operable to fetch program instructions from a sequence of memory locations;

an instruction decoder responsive to program instructions fetched by said instruction fetching circuit to control data processing operations specified by said program instructions; and

an execution circuit operable under control of said instruction decoder to execute said data processing operations; wherein

said instruction decoder is responsive to an execute block instruction to trigger fetching of a block of two or more program instructions by said instruction fetching circuit and execution of said block of two or more program instructions by said execution circuit, said block of two or more instructions containing a number of program instructions specified by a block length field within said executed block instruction and being stored at a memory location specified by a location field within said execute block instruction.

2. Apparatus as claimed in claim 1, wherein after execution of said block of two or more program instructions a return is made to a program instruction outside of said block of two or more program instructions.

3. Apparatus as claimed in any one of claims 1 and 2, wherein said return is to a program instruction immediately following said execute block instruction within said sequence of memory locations.

4. Apparatus as claimed in any one of claims 1, 2 and 3, wherein said location field is an offset field specifying said location of said block of two or more program instructions relative to a memory location of said execute block instruction.

5. Apparatus as claimed in any one of the preceding claims, comprising a program counter register operable to store an address indicative of a memory location of a program instruction being executed within said sequence of program instructions.

6. Apparatus as claimed in any one of the preceding claims, comprising a block counter register operable to store a block count value indicative of a location of a program instruction being executed within said block of two or more program instructions.

7. Apparatus as claimed in claims 5 and 6, wherein when executing said block of two or more program instructions, said program counter registers stores an address indicative of a memory location of said execute block instruction and said block counter register stores a block count value indicative of said program instruction location of a program instruction being executed within said block of two or more program instructions corresponding to said execute block instruction.

8. Apparatus as claimed in any one of claims 6 and 7, comprising an exception handling circuit operable upon occurrence of an exception during execution of said block of two or more instructions to store said block count value and upon completion of handling of said exception to restart execution of said block of two or more program instructions at a program instruction within said block of two or more instructions indicated by said block count value.

9. Apparatus as claimed in claims 5 and 8, wherein said exception handling circuit is operable to store said address of said execute block instruction upon occurrence of said exception and to restore said address of said execute block instruction to said program counter register upon said completion of handling of said exception.

10. Apparatus as claimed in claims 2 and 5, wherein upon completion of execution of said block of two or more program instructions said instruction decoder is operable to return processing to a program instruction following said execute block instruction as indicated by said program counter register.

11. A method for processing data, said method comprising the steps of:

fetching program instructions from a sequence of memory locations with an instruction fetching circuit;

controlling data processing operations specified by said program instructions with an instruction decoder; and

executing said data processing operations with an execution circuit controlled by said instruction decoder; wherein

said instruction decoder is responsive to an execute block instruction to trigger fetching of a block of two or more program instructions by said instruction fetching circuit and execution of said block of two or more program instructions by said execution circuit, said block of two or more instructions containing a number of program instructions specified by a block length field within said executed block instruction and being stored at a memory location specified by a location field within said execute block instruction.

12. A method as claimed in claim 11, wherein after execution of said block of two or more program instructions a return is made to a program instruction outside of said block of two or more program instructions.

13. A method as claimed in any one of claims 11 and 12, wherein said return is to a program instruction immediately following said execute block instruction within said sequence of memory locations.

14. A method as claimed in any one of claims 11, 12 and 13, wherein said location field is an offset field specifying said location of said block of two or more program instructions relative to a memory location of said execute block instruction.

15. A method as claimed in any one of claims 11 to 14, comprising storing within a program counter register an address indicative of a memory location of a program instruction being executed within said sequence of program instructions.

16. A method as claimed in any one of claims 11 to 15, comprising storing within a block counter register a block count value indicative of a location of a program instruction being executed within said block of two or more program instructions.

17. A method as claimed in claims 15 and 16, wherein when executing said block of two or more program instructions, said program counter registers stores an address

indicative of a memory location of said execute block instruction and said block counter register stores a block count value indicative of said program instruction location of a program instruction being executed within said block of two or more program instructions corresponding to said execute block instruction.

18. A method as claimed in any one of claims 16 and 17, comprising upon occurrence of an exception during execution of said block of two or more instructions storing said block count value and upon completion of handling of said exception restarting execution of said block of two or more program instructions at a program instruction within said block of two or more instructions indicated by said block count value.

19. A method as claimed in claims 15 and 18, wherein upon occurrence of said exception storing said address of said execute block instruction and restoring said address of said execute block instruction to said program counter register upon said completion of handling of said exception.

20. A method as claimed in claims 12 and 15, wherein upon completion of execution of said block of two or more program instructions said instruction decoder is operable to return processing to a program instruction following said execute block instruction as indicated by said program counter register.

21. A computer program product including a computer program operable to control a data processing apparatus having an instruction fetching circuit operable to fetch program instructions from a sequence of memory locations, an instruction decoder responsive to program instructions fetched by said instruction fetching circuit to control data processing operations specified by said program instructions, and an execution circuit operable under control of said instruction decoder to execute said data processing operations; said computer program including one or more an execute block instructions operable to trigger fetching of a block of two or more program instructions by said instruction fetching circuit and execution of said block of two or more program instructions by said execution circuit, said block of two or more instructions containing a number of program instructions specified by a block length

field within said executed block instruction and being stored at a memory location specified by a location field within said execute block instruction.

22. A computer program product as claimed in claim 21, wherein after execution of said block of two or more program instructions a return is made to a program instruction outside of said block of two or more program instructions.

23. A computer program product as claimed in any one of claims 21 and 22, wherein said return is to a program instruction immediately following said execute block instruction within said sequence of memory locations.

24. A computer program product as claimed in any one of claims 21, 22 and 23, wherein said location field is an offset field specifying said location of said block of two or more program instructions relative to a memory location of said execute block instruction.

25. A computer program product as claimed in any one of claims 21 to 24, wherein a program counter register stores an address indicative of a memory location of a program instruction being executed within said sequence of program instructions.

26. A computer program product as claimed in any one of claims 21 to 25, wherein a block counter register stores a block count value indicative of a location of a program instruction being executed within said block of two or more program instructions.

27. A computer program product as claimed in claims 25 and 26, wherein when executing said block of two or more program instructions, said program counter registers stores an address indicative of a memory location of said execute block instruction and said block counter register stores a block count value indicative of said program instruction location of a program instruction being executed within said block of two or more program instructions corresponding to said execute block instruction.

28. A computer program product as claimed in any one of claims 26 and 27, wherein upon occurrence of an exception during execution of said block of two or more instructions storing said block count value and upon completion of handling of said exception restarting execution of said block of two or more program instructions at a program instruction within said block of two or more instructions indicated by said block count value.

29. A computer program product as claimed in claims 25 and 28, wherein upon occurrence of said exception storing said address of said execute block instruction and restoring said address of said execute block instruction to said program counter register upon said completion of handling of said exception.

30. A computer program product as claimed in claims 22 and 25, wherein upon completion of execution of said block of two or more program instructions said instruction decoder is operable to return processing to a program instruction following said execute block instruction as indicated by said program counter register.

ABSTRACT

CONTROLLING EXECUTION OF A BLOCK OF PROGRAM INSTRUCTIONS WITHIN A COMPUTER PROCESSING SYSTEM

A data processing apparatus 2 is provided with an execute block instruction EMB which specifies a memory location of a block of program instructions to be executed as well as the length of that block of program instructions. When the end of that block of program instructions has been reached as tracked in response to the specified length value, a return to the main program flow is triggered. The instruction decoder 14 can include a block counter register 22 to keep track of the position within the block of program instructions being called. The block of program instructions are fetched by a prefetch unit 16 into the instruction pipeline 12 following the execute block instruction and are treated as having a program counter value corresponding to the execute block instruction whilst the block counter value keeps track of their separate positions within the block of program instructions.

[Figure 2]

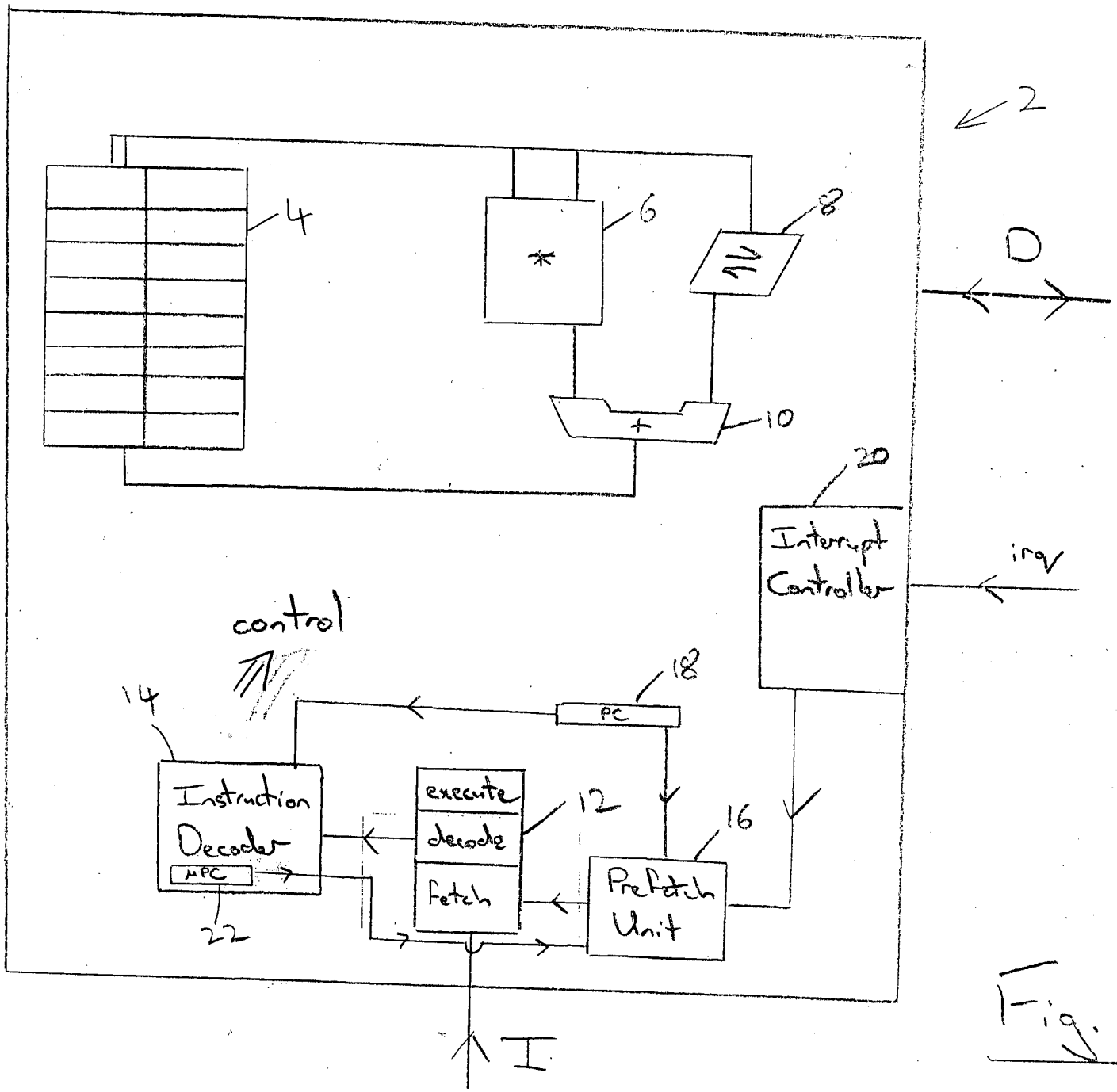

2
4
6
8
10
D
20
Interrupt Controller
irq
control
14
18
PC
Instruction Decoder
μPC
22
execute
decode
fetch
12
16
Prefetch Unit
I

Fig. 1

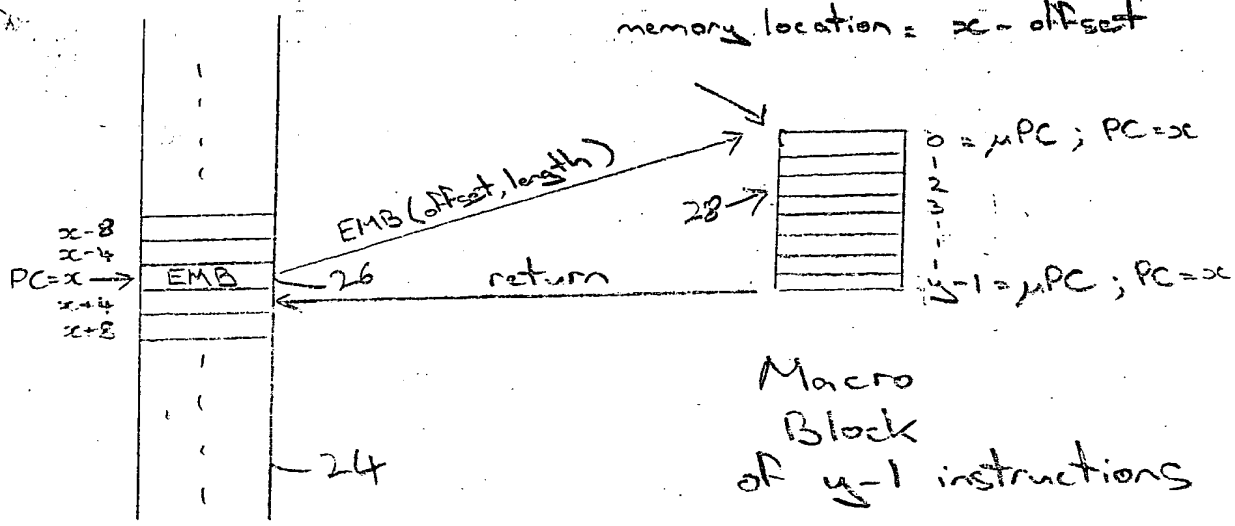

memory location = x - offset
EMB(offset, length)
EMB
PC=x
x-8
x-4
x+4
x+8
26
return
28
0 = μPC ; PC=x
1
2
3
y-1 = μPC ; PC=x
24
Macro Block of y-1 instructions
Main Code

Fig. 2

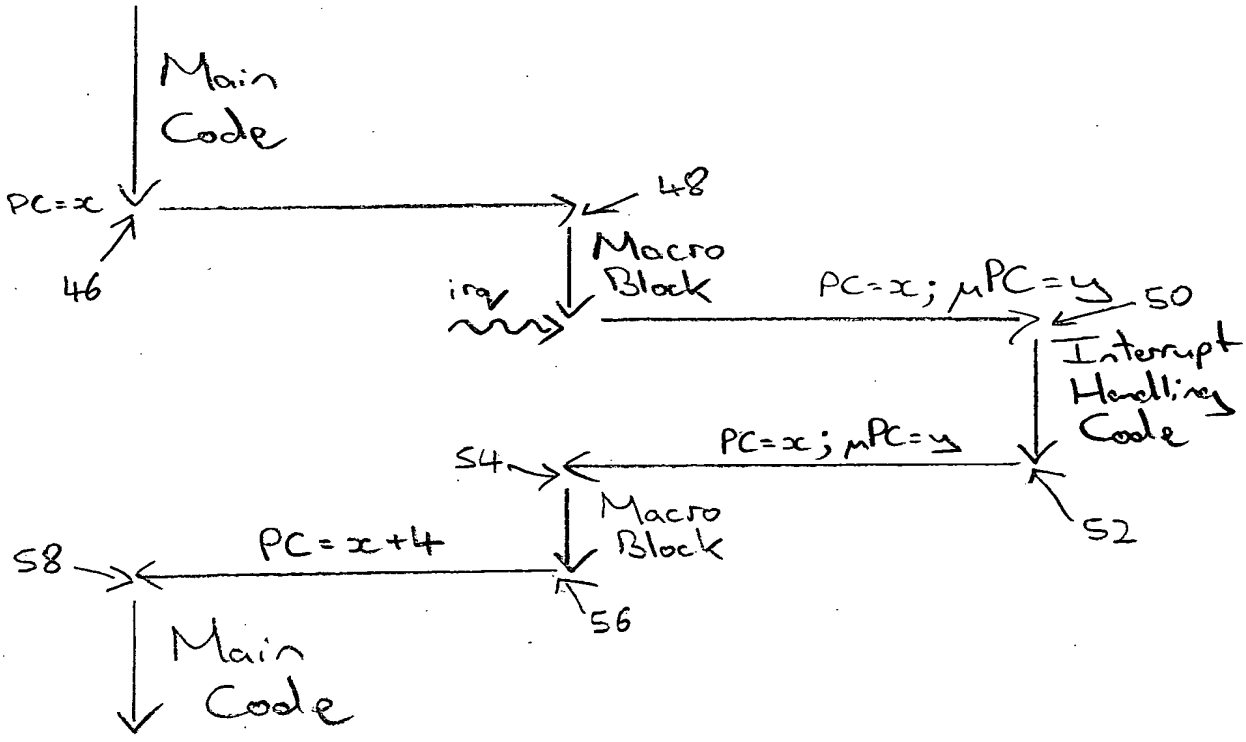

Main Code
PC=x
46
48
Macro Block
irq
PC=x; μPC=y
50
Interrupt Handling Code
52
PC=x; μPC=y
54
Macro Block
56
PC=x+4
58
Main Code

Fig. 4

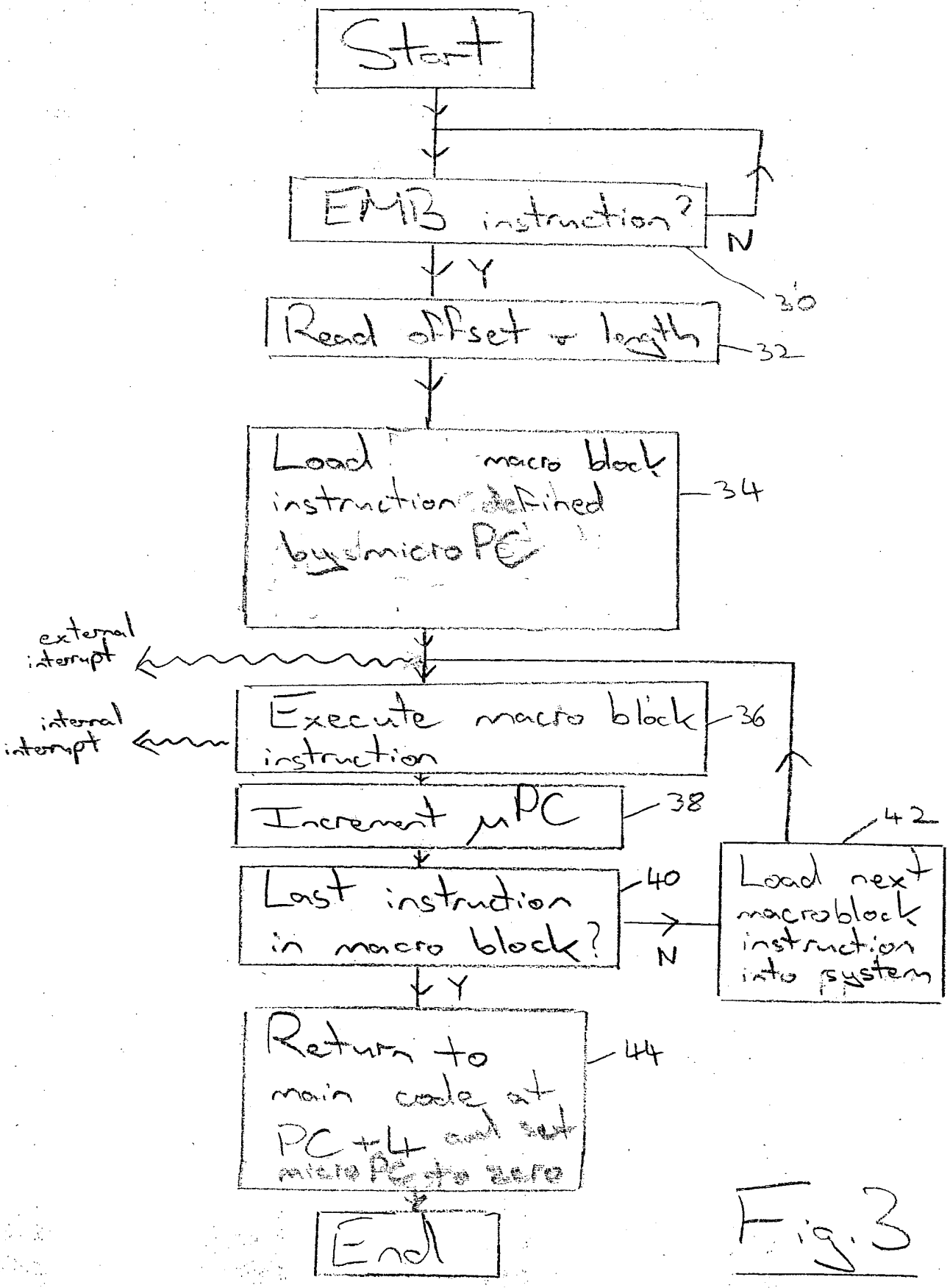

Start
EMB instruction?
N
Y
30
Read offset + length
32
Load macro block instruction defined by μmicro PC
34
external interrupt
internal interrupt
Execute macro block instruction
36
Increment μPC
38
Last instruction in macro block?
40
N
Y
Load next macroblock instruction into system
42
Return to main code at PC+4 and set micro PC to zero
44
End

Fig. 3

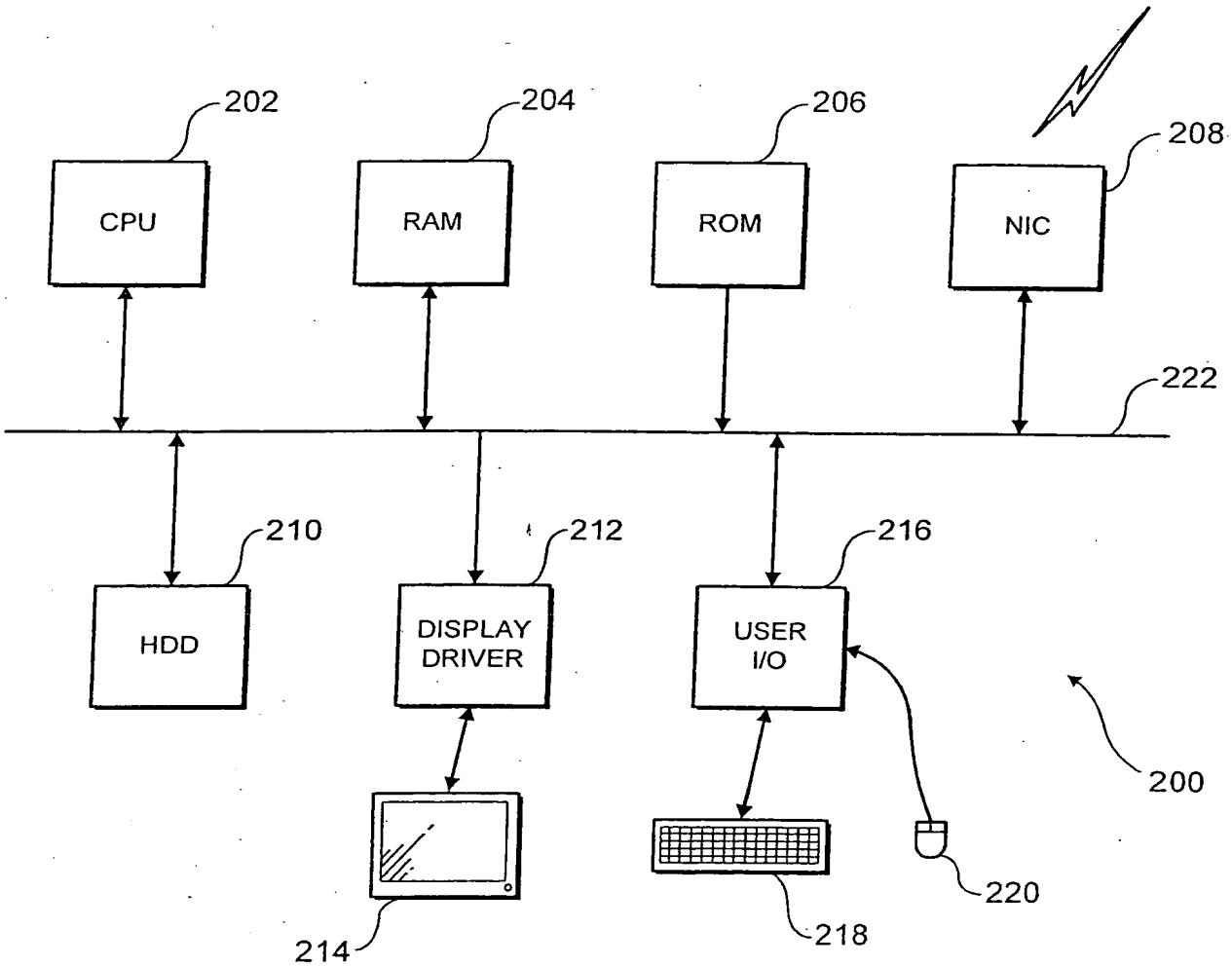

202
204
206
208
CPU
RAM
ROM
NIC
222
210
212
216
HDD
DISPLAY DRIVER
USER I/O
200
214
218
220

FIG. 5